(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 December 2001 (20.12.2001)

PCT

(10) International Publication Number
WO 01/97016 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US01/17263

(22) International Filing Date: 25 May 2001 (25.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/592,882 13 June 2000 (13.06.2000) US

(71) Applicant (for all designated States except US): INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants (for US only): HERNANDEZ, Thomas [US/US]; 4340 NW 147th Avenue, Portland, OR 97229 (US). STEWART, David [US/US]; 8140 SW Aralia Place, Beaverton, OR 97008 (US).

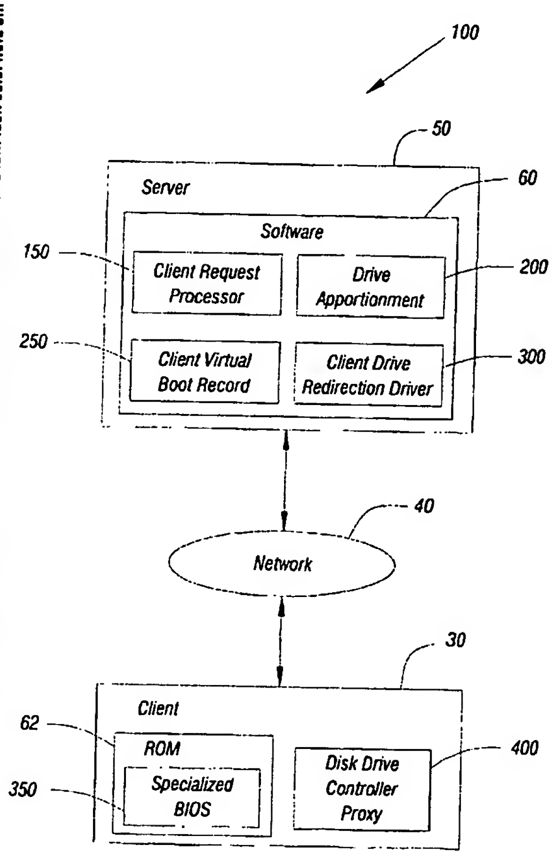
(74) Agent: TROP, Timothy, N.; Trop, Pruner & Hu, P.C., 8554 Katy Freeway, Suite 100, Houston, Tx Texas 77024 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: PROVIDING CLIENT ACCESSIBLE NETWORK-BASED STORAGE



(57) Abstract: A system for accessing storage on a server by clients on a network includes software on the server to process requests and allocate storage based on client needs. The clients include a special BIOS and a disk drive controller proxy. The BIOS retrieves operating system software, drivers, and other application software from the server. The disk drive controller proxy routes disk requests to the server for processing. In some embodiments, clients may share storage on the server.

WO 01/97016 A2

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

PROVIDING CLIENT ACCESSIBLE NETWORK-BASED STORAGE

Background

This invention relates to client-server environments and, more particularly, to diskless clients.

As the average selling price of personal computers declines, computer manufacturers often look for ways to more efficiently serve the market. In a client-server network environment, many redundancies may exist which present opportunities for computer manufacturers.

One possible redundancy is the use by several clients on a network of the same operating system and/or application programs. The vast majority of clients in a small business or educational environment have storage space that is identical and redundant. In particular, the storage space used for the operating system (such as Microsoft Windows) and the office productivity applications can take up to one gigabyte of space on a modern client. However, this is duplicated amongst the various clients.

The idea of a diskless client or diskless workstation is not new. However, with the advent of a 100 Mbit Ethernet Network Connection, a diskless client may become more practical. The server for these diskless clients may be designed to offer each client an amount of disk storage dedicated for that client. The space on the server would be the equivalent of a "C" drive (or main hard disk drive) for the client.

One way to optimize server storage for diskless clients is to divide the client's files into two categories: shared and private. Those files that are normally shared by all clients are loaded into a different directory structure than the private files. Microsoft used this implementation to support diskless clients in Windows 95. However, this implementation is problematic because most applications do not install and work properly when operating system files are loaded into more than one directory structure. In part because of this, Microsoft chose to drop the diskless support after Windows 95.

Other attempts to support diskless clients have been made. These methods tend to be highly complex and focus on using a higher-level file system protocol, such as the Network File System (NFS). However, particularly in low-end small business environments with very simple networks, such solutions may be too complex.

Thus, there is a continuing need to provide a diskless client-server environment that works seamlessly with operating system and other software.

Brief Description of the Drawings

Figure 1 is a block diagram of a system according to one embodiment of the invention;

Figure 2 is a flow diagram of the client request processor according to one embodiment of the invention;

Figure 3 is a block diagram of the drive apportionment software according to one embodiment of the invention;

Figure 4 is a detailed block diagram of the drive apportionment software according to one embodiment of the invention;

Figure 5 is a flow diagram of operation of the drive apportionment software according to one embodiment of the invention;

Figure 6 is a block diagram of the driver apportionment software using a cached according to one embodiment of the invention;

Figure 7 is a flow diagram of the specialized BIOS of the client according to one embodiment of the invention;

Figure 8 is a flow diagram showing operation of the client virtual boot record according to one embodiment of the invention;

Figure 9 is a flow diagram showing operation of the disk drive controller proxy according to one embodiment of the invention; and

Figure 10 is a block diagram of the system according to one embodiment of the invention.

Detailed Description

The following describes a system for providing network-based storage to clients on a network. For purposes of explanation, specific embodiments are set forth to provide a thorough understanding of the invention. However, it will be understood by one skilled in the art that the invention may be practiced without the particular details described herein. Further, although the embodiments are described in terms of hard disk drive storage media, the illustrated system may be applied to other non-volatile media, including, but not limited to compact disk read-only memories (CD ROMs), optical storage media, tape drives, flash memories, and option ROMs. Moreover, well-known elements, devices, process steps, and the like, are not set forth in detail in order to avoid obscuring the invention.

In accordance with the several embodiments described herein, a server system on a network may support a plurality of diskless clients. Where possible, software on the server, and thus storage space, is shared between the diskless clients, providing an efficient network. *Using specialized firmware on the client,* the client is able to communicate with software on the server to redirect disk accesses. By providing a proxy for the disk drive controller on the client, operating system and other software operate as though the client has a local hard disk drive.

In Figure 1, a system 100 includes a server 50 and a client 30, both connected to a network 40. The server 50 and the client 30 are both processor-based systems. In one embodiment, the server 50 and client 30 are connected to the network 40 using a fast Ethernet Local Area Network (LAN) protocol, which supports data transfer rates of 100 megabits per second.

In one embodiment, the server 50 includes a plurality of software 60. For example, a client request processor 150 receives network packets indicating hard disk drive requests from the client 30. The client request processor 150 translates these requests such that other software 60 on the server 50 may process them. The client request processor 150 further translates results from operations performed on the server 50 such that they may be transmitted back to the client

30. The client request processor 150, according to one embodiment, is described in connection with Figure 2, below.

The server software 60 further includes drive apportionment software 200. The drive apportionment software 200 receives hard disk drive access requests from the client request processor 150. The drive apportionment software 200 services the requests based upon a particular mapping of the hard disk drive on the server 50. The drive apportionment software 200, according to several embodiments, is described in more detail in connection with Figures 3 through 6, below.

The server 50 further includes one or more client virtual boot records 250. The client virtual boot record or records 250, in one embodiment, may be downloaded to the client 30 following a request by the client 30 to boot. In some embodiments, the server 50 includes more than one client virtual boot record 250, such that the various clients 30 on the network 40 may be flexibly configured. The client virtual boot record 250, according to one embodiment, is discussed in connection with Figure 8, below.

In one embodiment, the server 50 further includes a client drive redirection driver 300. The client drive redirection driver 300, like the client virtual boot record 250, is downloaded to the client 30 during initialization. The drive redirection driver 300 receives requests to access a hard disk drive, which is not present on the client 30. Once loaded on the client 30, the drive redirection driver 300 transmit the requests over the network 40 to the server 50, such that the requests may be serviced on the hard disk drive of the server 50. The client drive redirection driver 300, according to one embodiment, is also described in connection with Figure 8, below.

In addition to the software 60 on the server 50, the client or clients 30 include both a specialized basic input/output system (BIOS) 350 and a disk drive controller proxy 400. In one embodiment, the specialized BIOS 350 is part of a memory 62, such as a read-only memory (ROM). The specialized BIOS 350 assists the client 300 during power on such that the client 30 is connected to the network 40, receives the attention of the server 50, receives the virtual boot

record 250 from the server 50, and receives the drive redirection driver 300 from the server 50. Together, these operations permit the diskless client 30 to nevertheless service disk drive requests. The specialized BIOS 350, according to one embodiment, is discussed in connection with Figure 7, below.

The disk driver controller proxy 400, in one embodiment, causes software running on the client 30 to believe that a disk drive controller is present on the client 30. Thus, software operating on the client 30 sends a disk drive request to the disk drive controller proxy 400, just as the software would send a request to a real disk drive controller. However, in some embodiments, the disk drive controller proxy 400 may operate differently than a disk drive controller. Instead, in one embodiment, the disk drive controller proxy 400 encapsulates the disk requests in packets such that they may be transmitted over the network 40, and ultimately serviced by the drive apportionment software 200 of the server 50.

The disk drive controller proxy 400 may also receive packets from the network 40 and return the drive results to the requesting software. In one embodiment, the disk drive controller proxy 400 is hardware-based, embedded in a multi-purpose chip, such as a southbridge controller chip. In a second embodiment, the disk drive controller proxy 400 is firmware, stored in the ROM 62 of the client 30. The disk drive controller proxy 400, according to one embodiment, is discussed in more detail in connection with Figure 9, below.

The client request processor 150 may receive disk requests from the client 30 encapsulated as network packets, as well as transmit responses to the disk operations back to the client 30. In Figure 2, the operation of the client request processor 150, according to one embodiment, begins by receiving by a packet over the network 40 on behalf of the client 30 (block 152). In one embodiment, the packet is an Ethernet packet. However, the packet may be transmitted using other protocols.

From the packet, the client request processor 150 extracts a drive request from the encapsulated packet (block 154). The client request processor 150 then calls the drive apportionment software 200 (block 156). The drive apportionment software 200, in one embodiment, translates the drive request by the client 30

into a local storage request, that is, a request for the hard disk drive of the server 50.

After servicing the drive request, the client request processor 150 receives one or more results from the drive apportionment software 200 (block 158). Reversing the prior operations, the client request processor 150 encapsulates the drive results into packets (block 160), such that the packets may be transmitted over the network 40 (block 162). Thus, the operation of the client request processor 150, according to one embodiment, is complete.

When the drive apportionment software 200 is called by the client request processor 150, the request may be serviced in a number of ways. For example, the server 50 may allocate a distinct portion of its storage space to each requesting client 30 on the network 40. On a typical network, each client 30 may boot the same operating system or may use the same application software as another client on the network 40. Thus, according to one embodiment, the system 100 allocates server 50 storage with this common usage model in mind.

In Figure 3, according to one embodiment, the server 50 includes a non-volatile storage, such as a hard disk drive 80, including a common storage region 82, as well as private storage regions 84 for each of the clients 30 on the network 40. In a second embodiment, the non-volatile storage is an optical disk.

In Figure 3, the network 40 includes client 30a, client 30b, and client 30c. Accordingly, the hard disk drive 80 includes client private storage 84a, client private storage 84b, and client private storage 84c.

Further, the server 50 includes a memory 20, which stores an information database, such as a table, which keeps track of which client is using which portion of the hard disk drive 80. In one embodiment, the memory 20 stores a bitmap 22, one for each client 30, to supply this information. For each client 30a, 30b, and 30c, the memory 20 includes distinct bitmaps: a client bitmap 22a, a client bitmap 22b, and a client bitmap 22c. These bitmaps 22 assist the drive apportionment software 200 to determine where on the hard disk drive 80 to service requests from the client 30.

Each bitmap 22 is comprised of a plurality of bits. In one embodiment, each bit represents a sector of disk storage available to each client 30, as between the common storage 82 and the private storage 84. Further, in one embodiment, the location of the bit 12 in the bitmap 22 represents the location of the sectors in the common storage 82.

In Figure 4, the client bitmap 22a, according to one embodiment, is located in the memory 20 of the server 50, and includes a plurality of bits 12. In one embodiment, the number of bits 12 in the client bitmap 22a represents the number of sectors available to the client 30a, while the number of bits 12 set to a "1" represents the number of private sectors available to the client 30a.

In Figure 4, the common storage 82 of the server 50 includes a plurality of sectors 86. Each bit 12 of the client bitmap 22a initially represents each sector 86 of the common storage 82. However, in one embodiment, each time a bit 12 is set in the client bitmap 22a, a copy of the sector 86 is copied from the common storage 82 to the client private storage 84a as a private sector 88.

In addition to identifying the number of sectors available to the client 30a, the bitmap 22a further may identify the location of the requested sector 86 on the hard disk drive 80. For example, in one embodiment, a "0" in the bitmap 22a for bit 12 in the third position directs the drive apportionment software 200 to retrieve the third sector from the common storage 82 of the hard disk drive 80.

By contrast, a "1" in the bitmap 22a for the ninth bit, bit 12a in Figure 4, indicates that the sector to retrieve was originally in the ninth position, sector 86a in Figure 4, of the common storage 82. However, the sector 86a has previously been copied to the client private storage 84a as private sector 88a. Because bit 12a is the first non-zero occurrence in the client bitmap 22a, the first private sector 88a is retrieved from the client private storage 84a. Thus, the "1" in bit 12a of the bitmap 22a directs the drive apportionment software 200 to retrieve the first private sector 88a from the private storage 84a on behalf of the requesting client 30a. Programmers of ordinary skill in the art recognize this as but one of several possible implementations of the client bitmap 22a.

Thus, when all bits 12 in the bitmap 22a are set to "0", the drive apportionment software 200 retrieves all sector read requests from the common storage 82 of the hard disk drive 80. For some systems 100, the common storage area 82 may service the vast majority of accesses by the clients 30 on the network 40. For example, the loading of operating system and other application software typically involves only read operations. Accordingly, client 30a and client 30c may share identical operating system software located in the common storage area 82 of the hard disk drive 80.

Occasionally, the client 30 may perform write operations to its virtual storage. In one embodiment, when a sector write operation is requested by the client 30, the bit 12 corresponding to the requested sector 86 is set to a "1" (that is, if the bit 12 has not already been set). The relevant sector 86 is copied to the client private storage 84 as private sector 88, where the write operation may subsequently be performed. Once a write to a sector is made, all subsequent accesses to the sector are from the private storage 84 of the client 30.

In Figure 4, four bits of the client 30 bitmap 22a are set to a "1", 12a, 12b, 12c and 12d. Likewise, four sectors 86 of the common storage 82, 86a, 86b, 86c, and 86d, are copied to the client private storage 84a, as private sectors 88a, 88b, 88c, and 88d. Thus, in one embodiment, the number of bits 12 which are set to "1" in the client bitmap 22 represents the number of private sectors 88 located in the client private storage 84.

In Figure 5, the drive apportionment software 200, according to one embodiment, begins by receiving the hard disk drive 80 access request, such as from the client request processor 150, on behalf of the client 30 (block 202). In one embodiment, all read requests are initially serviced from the common storage 82 of the hard disk drive 80. However, once a sector 86 is copied as a private sector 88 and the private sector 88 is written to, subsequent read requests of the sector 86 are directed to the private storage 84 of the client 30 and the associated private sector 88. Accordingly, the drive apportionment software 200 determines whether a read or a write request is being made (diamond 204).

If a read request is made, the drive apportionment software 200, running on the server 50, accesses the bitmap 22 of the client 30 from the memory 20 of the server 50. The bit 12 for the requested sector is tested (diamond 206). If the relevant bit 12 is set, a prior write operation to the private sector 88 may be presumed. If the bit 12 is not set, the requested sector 86 is retrieved from the common storage 82 of the server 50 (block 208). The requested sector 86 is then sent to the client request processor 150 (block 210). From there, according to one embodiment, the client request processor 150 sends the result across the network 40 to the client 30.

If the relevant bit 12 of the client's bitmap 22 is set (diamond 206), then, although this is a read request, the private sector 88 is retrieved from the private storage 84 of the client 30 (block 212). The retrieved private sector 88 is then sent to the client request processor 150, where it is packetized and transmitted to the client 30 over the network 40.

To process write requests (diamond 204), the drive apportionment software 200 reads the bitmap 22 of the client 30 to determine if the relevant bit 12 is set (diamond 214). If so, the requested sector is already in the private storage 84 of the client 30 as private sector 88. Accordingly, the private sector 88 is identified (block 220), then a write operation is performed on the private sector 88 in the client private storage 84 (block 222).

If, however, the relevant bit 12 is not set in the client bitmap 22, the requested sector 86 is copied from the common storage 82 and stored as private sector 88 in the private storage 84 of the client 30 by the drive apportionment software 200 (block 216). The relevant bit 12 is then set in the client bitmap 22 (block 218). This ensures that subsequent retrievals are to the private sector 88 in the private storage 84 of the client 30, not from the sector 86 in the common storage 82. The requested private sector 88 is then written to in the private storage 84 of the hard disk drive 80 of the server 50 (block 222). Thus, the operation of the drive apportionment software 200, according to one embodiment, is complete.

In some embodiments, much of the frequently used portions of the common storage 82 of the server 50 may be stored in a cache, such as a memory cache. In this way, read requests from the client 30 may be honored from in-memory cache, rather from the common storage 82 of the hard disk drive 80. In other embodiments, a processor cache is used to store commonly used portions of the common storage 82. The use of a cache may further speed operations such as booting or launching a productivity application by the client 30.

In Figure 6, the server 50 includes the common storage 82, just as in Figure 4. The server 50 further includes a cache bitmap 24, which, like the client bitmaps 22, may be located in the memory 20 of the server 50.

The memory 20 may further include a server cache 44. Like the client private storage 84 of Figure 4, the server cache may be used to transfer the sectors 86 from the common storage 82 as new sectors 92, such as for sectors which are frequently used. Unlike the client private storage 84, which is located on the hard disk drive 80, the server cache 44 is located in the memory 20. Thus, in one embodiment, accesses to the sectors 92 in the server cache 44 may be serviced at a substantially higher speed than those to the hard disk drive 80 of the server 50. In Figure 6, the frequently used sectors, 86e through 86l, as indicated by the bits 12e through 12l in the cache bitmap 24, are accessible in the server cache 44 as sectors 92e through 92l.

In another embodiment, the client bitmaps 22 as well as the cache bitmap 24 may identify blocks different from a sector in size for movement to the client private storage 84 or the server cache 44. A hard disk drive sector is typically 512 bytes in length. However, modern file systems often deal with block sizes of seven sectors. The larger block size is used so that the file system runs more efficiently. So, for example, if a single sector is changed in the middle of a seven sector file system block, according to one embodiment, all seven sectors may be copied into the private storage 84 of the client 30 (or into the server cache 44). Accordingly, the bit 12 corresponding to the seven sector block may be set.

In yet another embodiment, the client bitmaps 22 and the cache bitmap 24 may be reset such that all bits 12 in the bitmap 22 are cleared to "0". For a client

bitmap 22, this state may be identified as a default state of the client 30. The ability to reset the client 30 to a default state may be particularly useful in environments where the user of the client 30 changes. For example, in an educational setting, a student user may change on the client 30 every six months to a year.

Looking back to Figure 1, the client request processor 150 and the drive apportionment software 200, in one embodiment, are not run until the client 30 establishes a connection to the server 50. The specialized BIOS 350 of the client 30 engages the server 50, such that the client 30 may receive the virtual boot record 250 as well as the one or more client drive redirection driver 300 from the server 50. Additionally, the disk drive controller proxy 400, "fakes out" any application software executed by the client 30, such that hard disk drive operations may be successfully performed on the client 30 even though the client 30 itself has no hard disk drive.

In short, according to one embodiment, the specialized BIOS 350 of the client 30, soon after the client 30 powers on, retrieves the virtual boot record 250 and the client drive redirection driver 300 from the server 50. Once received, the virtual boot record 250 is executed on the client 30. During this time, any hard disk drive accesses, which are attempted by the virtual boot record 250, are redirected by the drive redirection driver 300 such that the disk drive controller proxy 400 of the client 30 is instead accessed. Then, the disk drive controller proxy 400 performs disk operations by accessing the hard disk drive 80 of the server 50.

As illustrated in Figure 2, the packets comprising disk requests sent by the disk drive controller proxy 400 to the server 50 are handled by the client request processor 150. The disk drive controller proxy 400 operates from the perspective of the requesting boot record 250 (or any other application software), as though the drive operations are performed locally. Thus, the client 30 may support operating system and other software, some of which depend upon the presence of a hard disk drive, while, in fact, having no hard disk drive.

In Figure 7, the operation of the specialized BIOS 350, according to one embodiment, starts by connecting the client 30 to the network 40, for access to the server 50 (block 352). The specialized BIOS 350 next broadcasts a request for the server 50 (block 354). In one embodiment, the client 30 and the server 50 operate as part of a pre-boot execution environment (PXE). The PXE Specification (Version 2.1, September 20, 1999) is available from Intel Corporation, Santa Clara, California. The client 30 initiates the PXE protocol by broadcasting a particular command that identifies the request as coming from a client that implements the PXE protocol. After several intermediate steps, the server 50 sends the client 30 a list of appropriate boot servers. The client 30 then discovers a boot server of the type selected and receives the name of an executable file on the chosen boot server.

In one embodiment, the executable file received by the client 30 is the client virtual boot record 250 (block 356). Further, the specialized BIOS 350 retrieves the client drive redirection driver 300, also from the server 50 (block 358). Finally, the specialized BIOS 350 executes the virtual boot record 250 (block 360). For example, the specialized BIOS 350 may perform a BIOS function INT19h, in order to execute the virtual boot record 250, which is loaded in a client memory. Thus, the operation of the specialized BIOS 350, according to one embodiment, is complete.

Now that the client virtual boot record 250 is loaded in the client memory, the virtual boot record 250 may be executed. Note that when the boot record is located on the hard disk drive, it is typically loaded into memory and executed. Although retrieved from a remote hard disk drive, the virtual boot record 250 thus also operates in this manner.

In Figure 8, the operation of the client virtual boot record 250, according to one embodiment, begins by issuing a command to load an operating system core (block 252). In the prior art, the operating system is typically loaded by issuing INT13h functions for retrieving the operating system from the hard disk drive. Here, instead, the drive redirection driver 300, received from the server 50 by the specialized BIOS 350, sends a request to the disk drive controller proxy 400 (block

254). In other embodiments, instead of using a driver to intercept the INT13h functions, the specialized BIOS 350 may itself redirect all INT13h functions to the disk drive controller proxy 400.

To process the request, in one embodiment, the disk drive controller proxy 400 retrieves the operating system from the server 50, using a protocol as described in connection with Figure 9, below (block 256). The operating system core is then loaded into the client memory (block 258). Once loaded, the operating system is executed (block 260). For all subsequent disk drive requests (e.g., run time requests), the drive redirection driver 300 may service the requests (block 262). Thus, according to one embodiment, the operation of the client virtual boot record 250 is complete.

The disk drive controller proxy 400, invoked by the client virtual boot record 250 may look likes a normal disk drive controller to software running on the client 30, in one embodiment. This means that the disk drive controller proxy 400 receives disk access requests. The operation of disk drive controller proxy 400, according to one embodiment, is described in connection with Figure 9.

First, the disk drive controller proxy 400 receives a request from operating system or other software loaded on the client 30, to access a hard disk drive (block 402). The disk drive controller proxy 400 encapsulates this disk request into packets suitable for transmission across the network 40 (block 404).

In one embodiment, the client 30 employs a protocol for requesting sector accesses across the network 40 to the server 50. The protocol includes a private command and reply structure, including a command header section, a data content section, and an error handling section. Because the client request processor 150 is available for receiving the requests by the server 50, the commands from the disk drive controller proxy 400 may be simple, like "read sectors a, b, c, d" or "write sectors e, f, g, h".

Next, according to one embodiment, the disk drive controller proxy 400 sends the packet with the encapsulated disk request over the network 40 to the server 50 (block 406). The disk drive controller proxy 400 then waits for a response from the server 50.

At some point, the disk drive controller proxy 400 receives the response packet, including the disk result, from the client request processor 150 of the server 50 (block 408). The disk drive controller proxy 400 next de-encapsulates the drive result from the packet as drive result data (block 410). The result data may then be stored in the memory of the client 30 (block 412), such that the data may be retrieved by the requesting application. Thus, the operation of the disk drive controller proxy 400, according to one embodiment, is complete.

Referring to Figure 10, the server 50 includes a processor 10, and the memory 20, including the cache 44, as described in connection with Figure 6, above. The processor 10 may comprise a microcontroller, an X86 microprocessor, an Advanced RISC Machine (ARM) microprocessor, or a Pentium-based microprocessor, to name a few. The memory 20 may comprise various types of random access memories, such as dynamic random access memories (DRAMs), synchronous DRAMs (SDRAMs), static RAMs (SRAMs), single in-line memory modules (SIMMs), or double in-line memory modules (DIMMs), to name a few.

Both the processor 10 and the memory 20 are connected via a system bus 14. In one embodiment, a bridge chip 16 is also coupled to the system bus 14, such as for including a second bus 18, such as a peripheral component interconnect (PCI) bus 18. The PCI specification is available from the PCI Special Interest Group, Portland, Oregon 97214.

In one embodiment, the bridge chip 16 further supports the hard disk drive 80, including the software 60 as shown in connection with Figure 1. The PCI bus 18 may also support a network interface card (NIC) 42, for connecting the server 50 to the network 40.

The client 30 is also a processor-based system, including a processor 58, a memory 64, and the ROM 62, each coupled to a system bus 56. Like the processor 10 on the server 50, the processor 58 may include any of a variety of processors. Also like the server 50, the memory 64 may be any of a variety of random-access memories.

As in Figure 1, the ROM 62 includes the specialized BIOS 350. The ROM 62 may be a programmable ROM (PROM), an erasable programmable ROM (EPROM), and electrically erasable PROM, and so on.

Like the server 50, the client 30 includes a bridge chip 52 coupled between the system bus 56 and a PCI bus 48. The PCI bus 48 may also support a NIC card 46, for connection to the network 40. In one embodiment, the disk drive controller proxy 400, as in Figure 1, may be part of the multi-function bridge chip 52. In other embodiments, the disk drive controller proxy 400 may be a separate and distinct component, coupled to the system bus 56 or the PCI bus 48. The embodiment of Figure 10 thus represents one of many possible implementations of the illustrated system 100.

Thus, in some embodiments, a system for supporting diskless clients exploits the redundant needs of the clients on a network. Where possible, the client may share accesses to the hard disk drive of the server. The clients nevertheless may maintain private storage on the server, where needed. In some embodiments, the client includes a specialized BIOS for establishing connection to the server and downloading an operating system. In some embodiments, the client further includes a drive controller proxy for re-routing drive requests to the server. The server likewise includes specialized software, for servicing the remote requests, for allocating drive space to the various clients, and for downloading operating system and driver software to the client, in some embodiments.

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method, comprising:

generating on a client a request to access a non-volatile storage;

receiving the request by a non-volatile storage proxy on the client; and

accessing the non-volatile storage on a server by the non-volatile storage proxy.

2. The method of claim 1, accessing the non-volatile storage on a server by the non-volatile storage proxy further comprising:

encapsulating the request into a packet suitable for transmission over a network; and

sending the packet to a server connected to the network.

3. The method of claim 2, further comprising:

receiving the packet by the server;

de-encapsulating the request from the packet; and

producing a result based upon the request.

4. The method of claim 3, producing a result based upon the request further comprising:

identifying a block of the non-volatile storage on the server;

performing an operation on the block based upon the request; and

receiving in the non-volatile storage proxy a result from the server.

5. The method of claim 4, identifying a block of the non-volatile storage on the server further comprising:

accessing a table associated with the client in a volatile memory of the server; and

finding an entry in the table associated with the block.

6. The method of claim 4, receiving in the non-volatile storage proxy a result from the server further comprising:

receiving a second packet encapsulating the result produced by the server; and

de-encapsulating the result from the second packet.

7. The method of claim 5, further comprising:

reviewing the entry associated with the block;

identifying the block as having previously been written to by the client; and

retrieving the block from a first portion of the non-volatile storage on the server wherein the first portion is associated with the client.

8. The method of claim 5, further comprising:

reviewing the entry associated with the block;

identifying the block as not having previously been written to by the client; and

retrieving the block from a common portion of the non-volatile storage on the server.

9. The method of claim 5, further comprising:

storing the result in a volatile memory of the client.

10. A system comprising:

a server, comprising a non-volatile storage;

a network coupled to the server; and

a client coupled to the network, the client including a non-volatile storage proxy that receives a storage access request and forwards the storage access request to the server over the network.

11. The system of claim 10, wherein the non-volatile storage proxy forwards the storage access request to the server over the network by:

encapsulating the storage access request into a packet; and
sending the packet over the network to the server.

12. The system of claim 11, wherein the server further:

receives a packet from the client on the network;
extracts the storage access request from the packet; and
accesses the non-volatile storage in response to the storage access request.

13. The system of claim 12, wherein the server further:

produces a result associated with accessing the non-volatile storage;
encapsulates the result in a second packet; and
sends the second packet over the network to the client.

14. The system of claim 13, wherein the non-volatile storage proxy further:

receives the second packet from the client over the network;
de-encapsulates the result from the second packet; and
sends the result to a memory on the client.

15. The system of claim 10, wherein the client includes no non-volatile storage.

16. The system of claim 10, wherein the client further comprises a software program to:

broadcast a server request on the network;
retrieve a virtual boot record from the server; and
execute the virtual boot record.

17. A system, comprising:

a processor;

a network interface for connecting the system to a network; and

a non-volatile storage proxy coupled to the processor, wherein the non-volatile storage proxy:

receives a non-volatile storage access request;

encapsulates the non-volatile storage access request into a packet; and

sends the packet out over the network interface to a server.

18. The system of claim 17, wherein the non-volatile storage proxy further:

receives a second packet from the server; and

de-encapsulates a result from the second packet.

19. The system of claim 18, wherein the result is produced by the server following an access to a non-volatile storage located on the server in response to the request received by the server from the system.

20. The system of claim 17, further comprising:

a non-volatile memory storing a software program executed during power-on of the system, wherein the program:

broadcasts a request for the server on the network;

retrieves a virtual boot record from the server;

stores the virtual boot record in the memory; and

executes the virtual boot record.

21. The system of claim 20, wherein the program further:

retrieves a redirection driver from the server;

stores the redirection driver in the memory; and

loads the redirection driver.

22. The system of claim 21, wherein the drive redirection driver sends the request to access the non-volatile storage to the non-volatile storage proxy.

23. An article comprising a medium storing a software program that, upon execution, causes a processor-based system to:

generate on the processor-based system a request to access a non-volatile storage;

receive the request by a non-volatile storage proxy on the processor-based system; and

access the non-volatile storage on a server by the non-volatile storage proxy.

24. The article of claim 23, further storing software that, upon execution, causes a processor-based system to access the non-volatile storage on a server by the non-volatile storage proxy by:

encapsulating the request into a packet suitable for transmission over a network; and

sending the packet to a server connected to the network.

25. The article of claim 24, further storing software that, upon execution, causes a processor-based system to:

receive the packet by the server;

de-encapsulate the request from the packet; and

produce a result based upon the request.

26. The article of claim 25, further storing software that, upon execution, causes a processor-based system to produce a result based upon the request by:

identifying a block of the non-volatile storage on the server;

performing an operation on the block based upon the request; and

receiving in the non-volatile storage proxy a result from the server.

27. The article of claim 26, further storing software that, upon execution, causes a processor-based system to identify a block of the non-volatile storage on the server by:

accessing a table associated with the processor-based system in a volatile memory of the server; and

finding an entry in the table associated with the block.

28. The article of claim 27, further storing software that, upon execution, causes a processor-based system to:

review the entry associated with the block;

identify the block as having previously been written to by the processor-based system; and

retrieve the block from a first portion of the non-volatile storage on the server wherein the first portion is associated with the processor-based system.

29. The article of claim 27, further storing software that, upon execution, causes a processor-based system to:

review the entry associated with the block;

identify the block as not having previously been written to by the processor-based system; and

retrieve the block from a common portion of the non-volatile storage on the server.

30. The article of claim 26, further storing software that, upon execution, causes a processor-based system to receive in the non-volatile storage proxy a result from the server by:

receiving a second packet encapsulating the result produced by the server; and

de-encapsulating the result from the second packet.

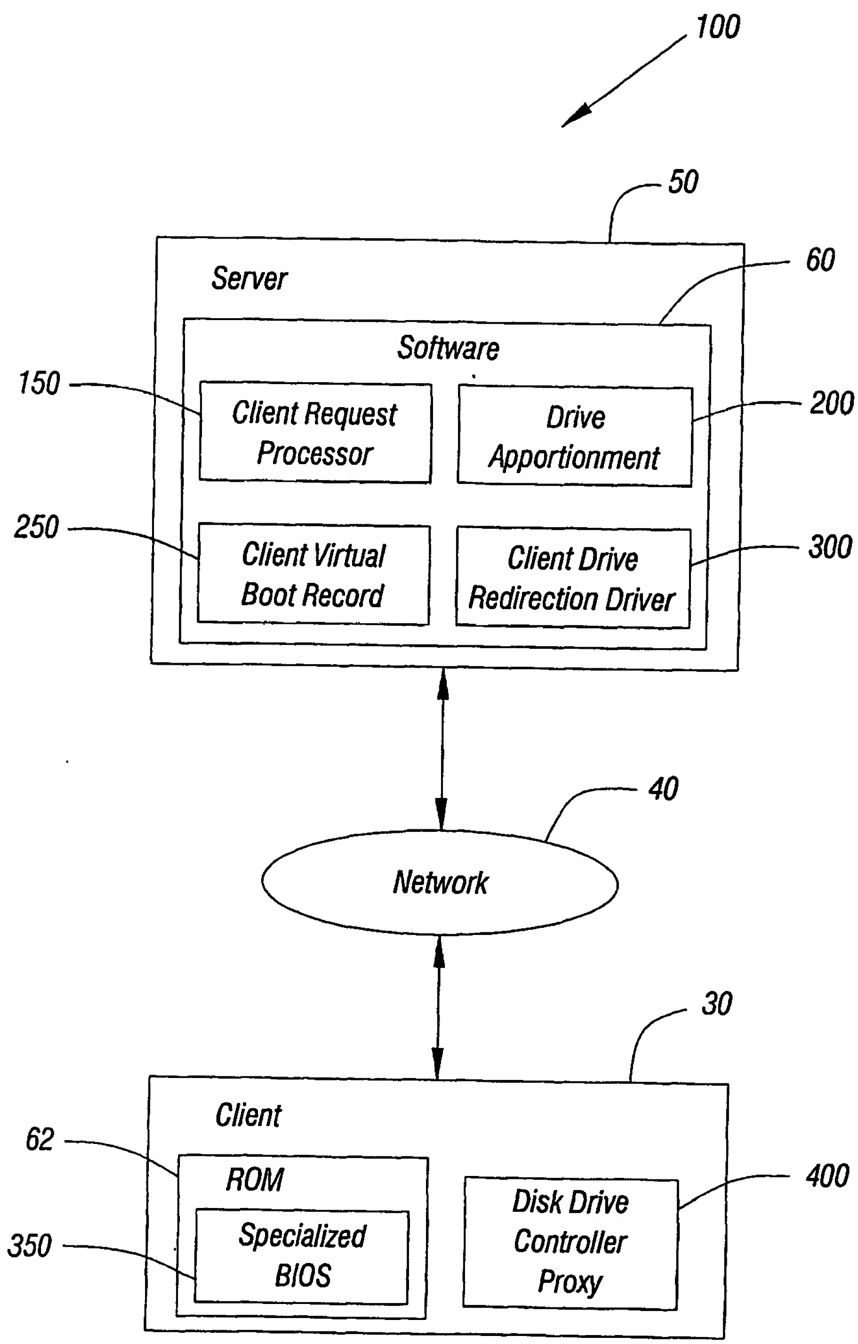

100
50
Server
60
Software
150
Client Request Processor
Drive Apportionment
200
250
Client Virtual Boot Record
Client Drive Redirection Driver
300
40
Network
30
Client
62
ROM
Specialized BIOS
350
Disk Drive Controller Proxy
400

Figure 1

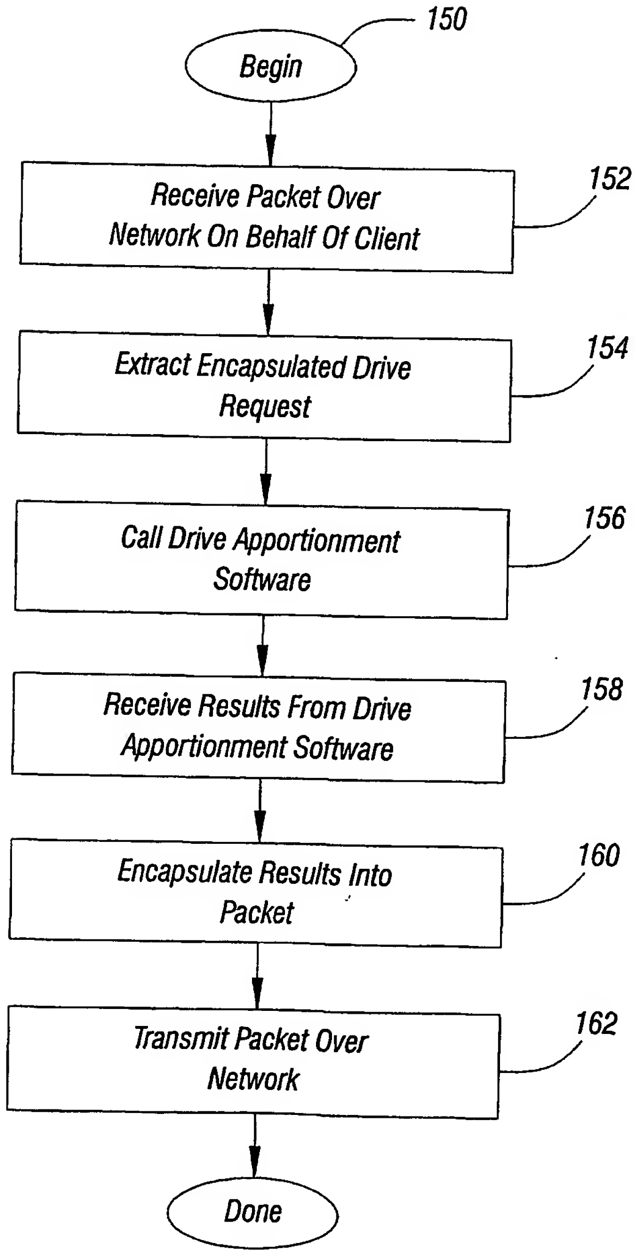

150
Begin
Receive Packet Over Network On Behalf Of Client
152
Extract Encapsulated Drive Request
154
Call Drive Apportionment Software
156
Receive Results From Drive Apportionment Software
158
Encapsulate Results Into Packet
160
Transmit Packet Over Network
162
Done

Figure 2

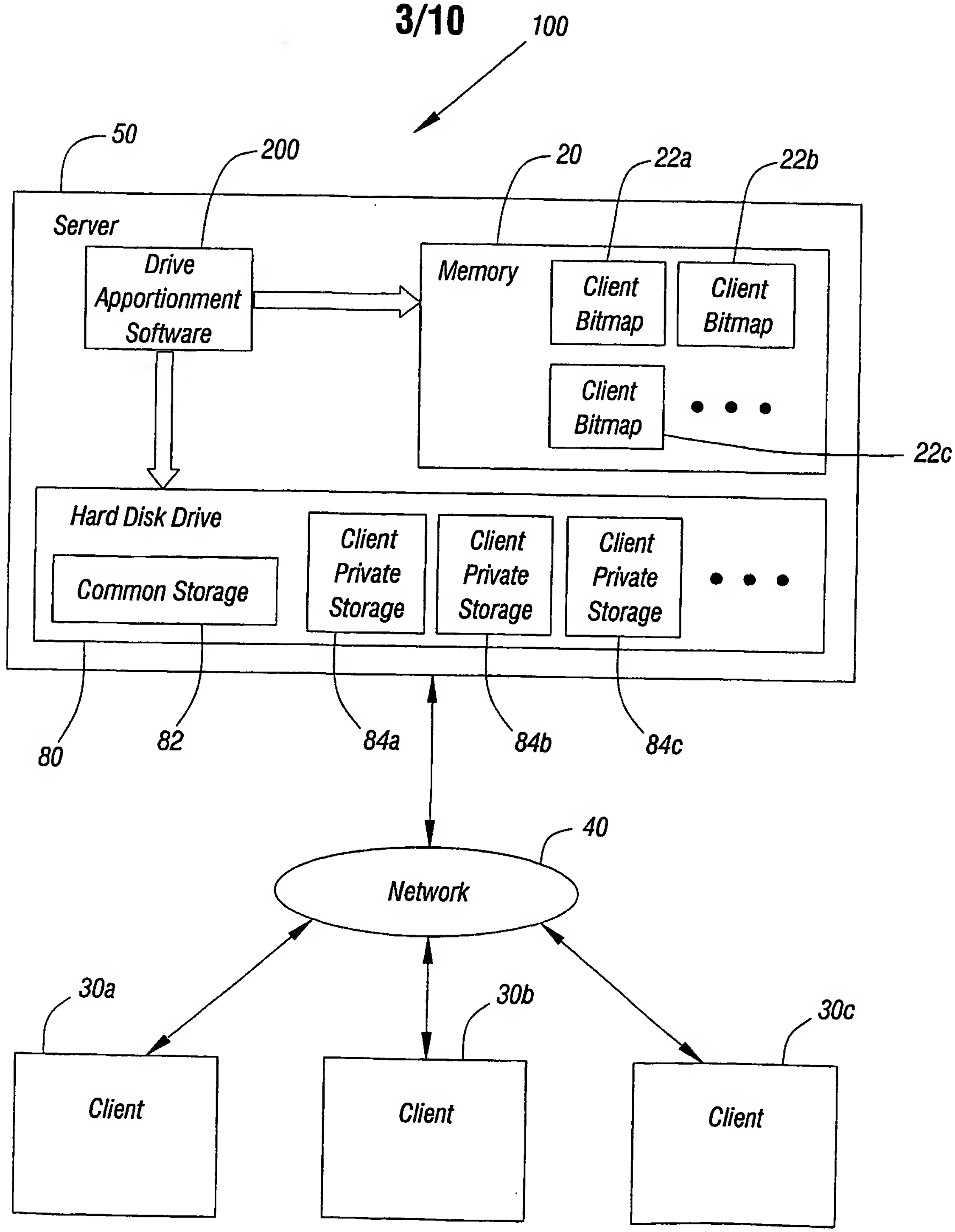

100
50
200
20
22a
22b
Server
Drive Apportionment Software
Memory
Client Bitmap
Client Bitmap
Client Bitmap
22c
Hard Disk Drive
Common Storage
Client Private Storage
Client Private Storage
Client Private Storage
80
82
84a
84b
84c
40
Network
30a
30b
30c
Client
Client
Client

Figure 3

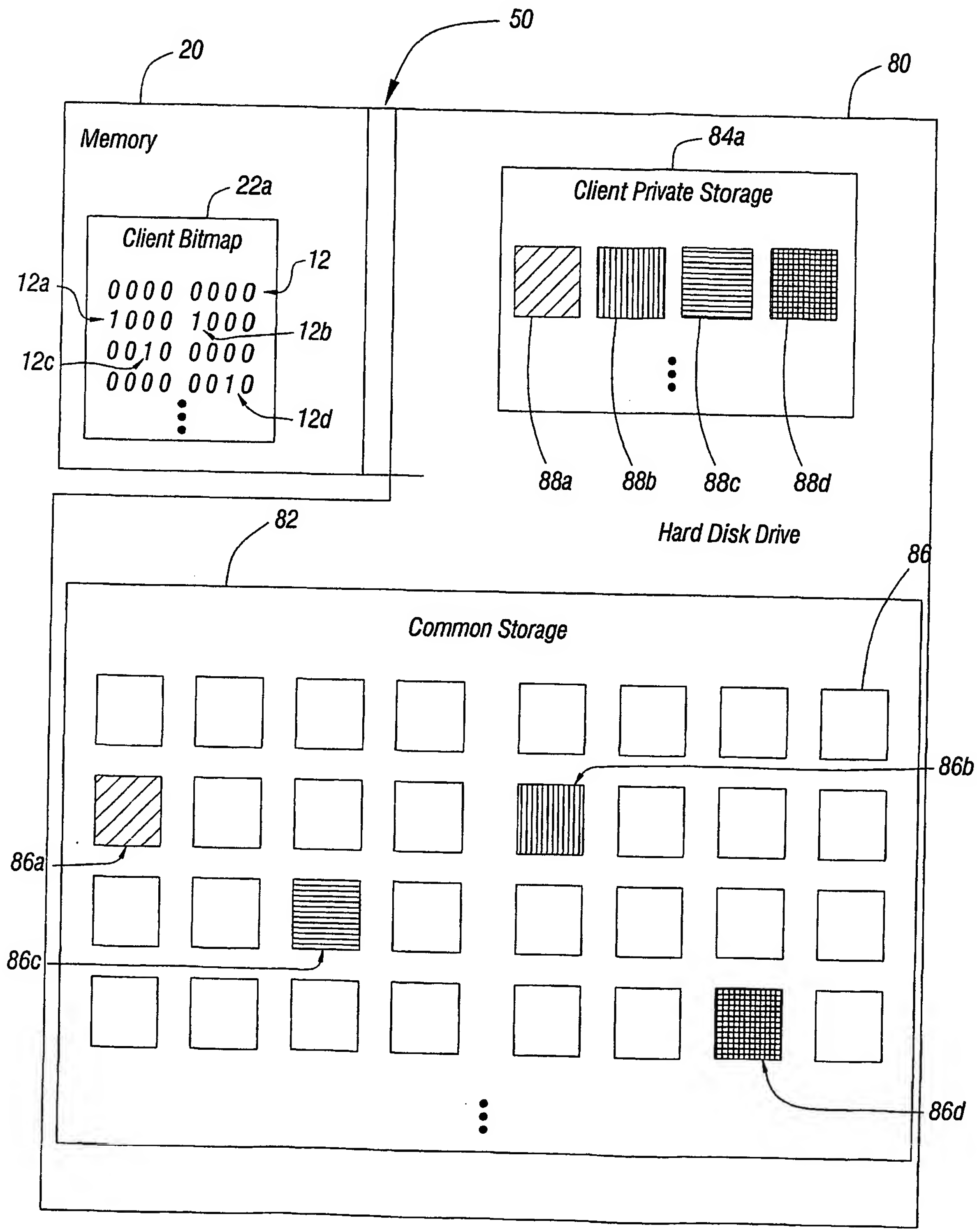

50
20
80
Memory
22a
Client Bitmap
12
12a
0000 0000
1000 1000
0010 0000
0000 0010
12b
12c
12d
84a
Client Private Storage
88a
88b
88c
88d
Hard Disk Drive
82
86
Common Storage
86a
86b
86c
86d

Figure 4

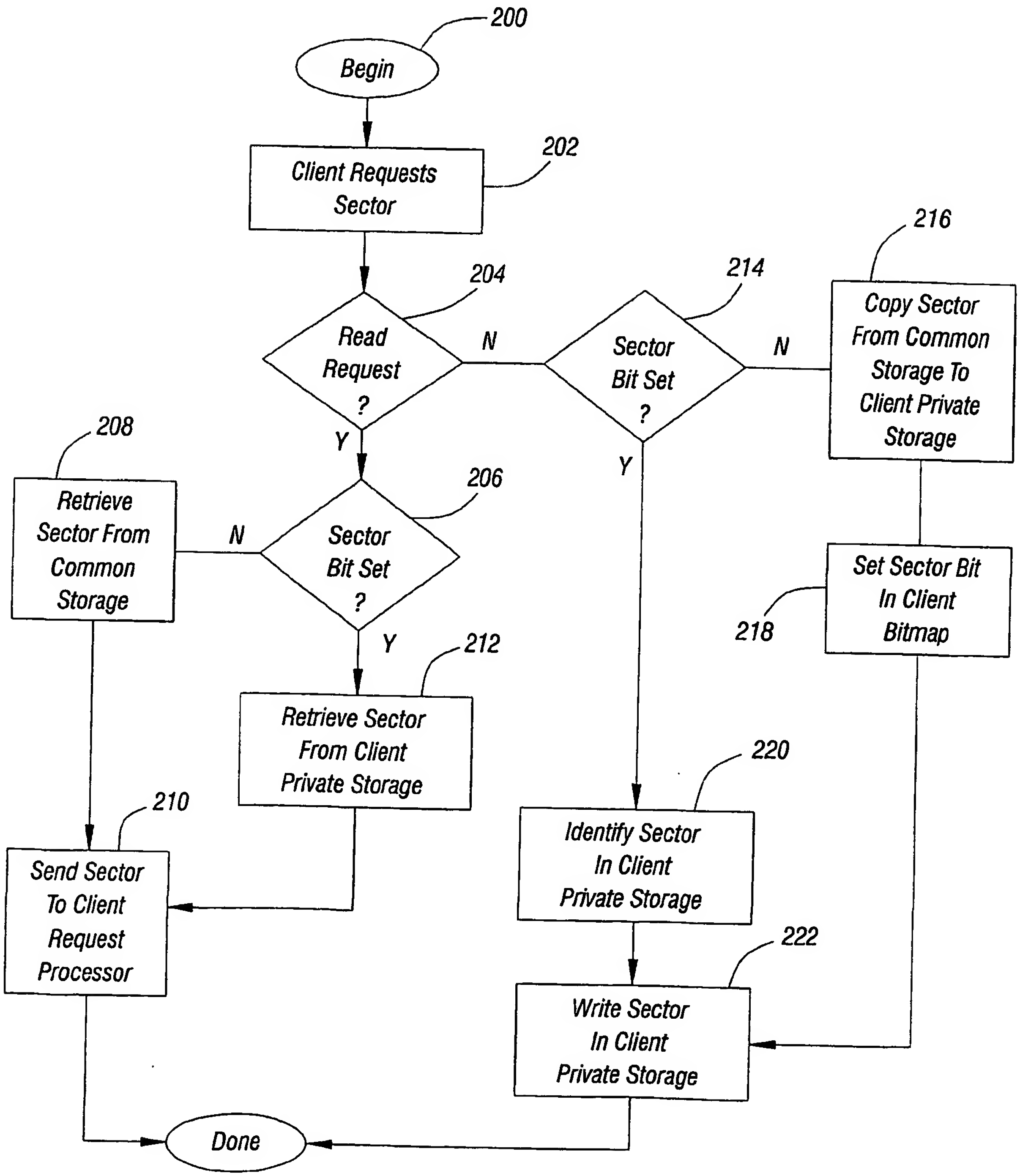

200
Begin
202
Client Requests Sector
204
Read Request ?
N
Y
214
Sector Bit Set ?
N
Y
216
Copy Sector From Common Storage To Client Private Storage
208
Retrieve Sector From Common Storage
N
206
Sector Bit Set ?
Y
212
Retrieve Sector From Client Private Storage
218
Set Sector Bit In Client Bitmap
220
Identify Sector In Client Private Storage
210
Send Sector To Client Request Processor
222
Write Sector In Client Private Storage
Done

Figure 5

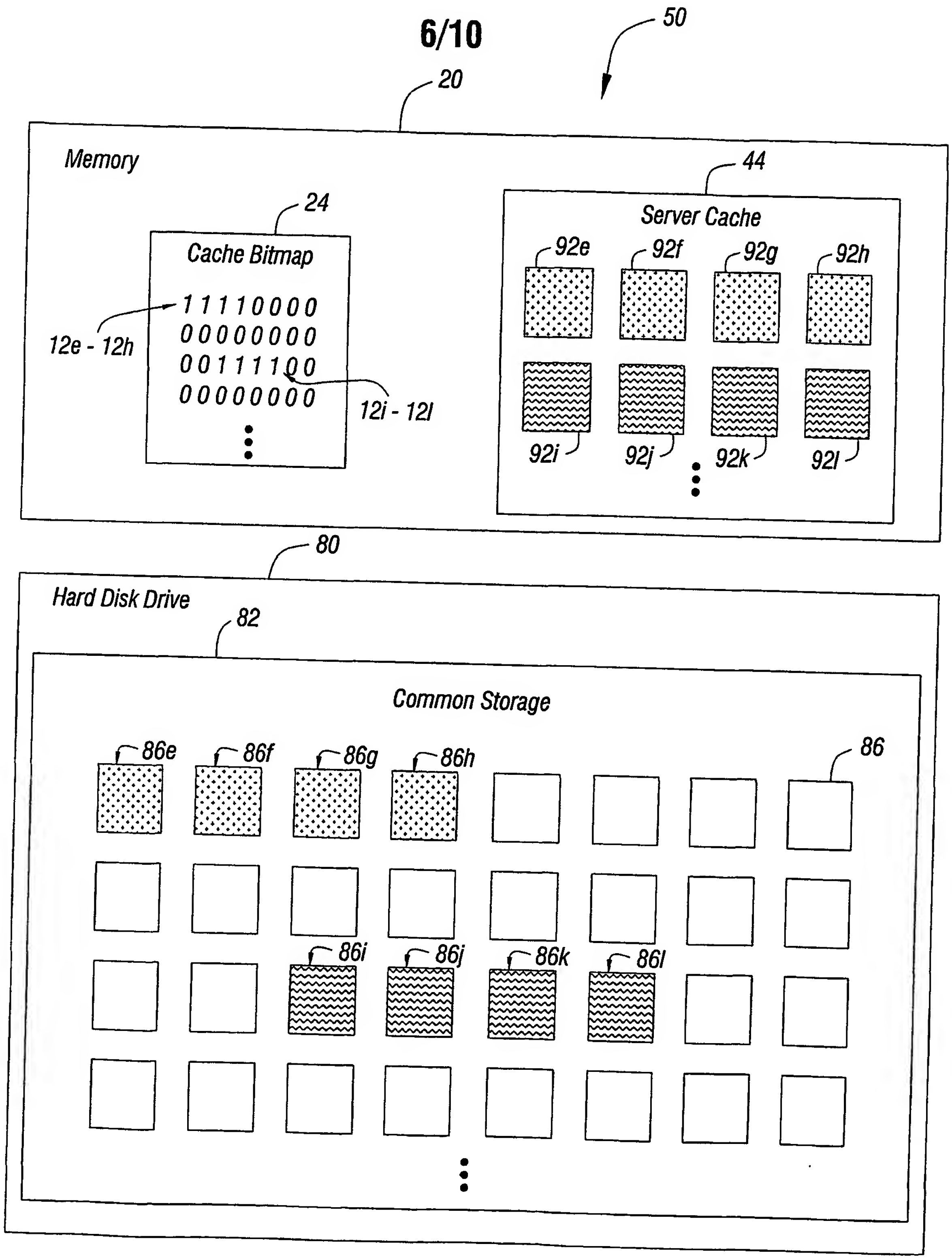

50
20
Memory
24
Cache Bitmap
1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
12e - 12h
12i - 12l
44
Server Cache
92e
92f
92g
92h
92i
92j
92k
92l
80
Hard Disk Drive
82
Common Storage
86e
86f
86g
86h
86
86i
86j
86k
86l

Figure 6

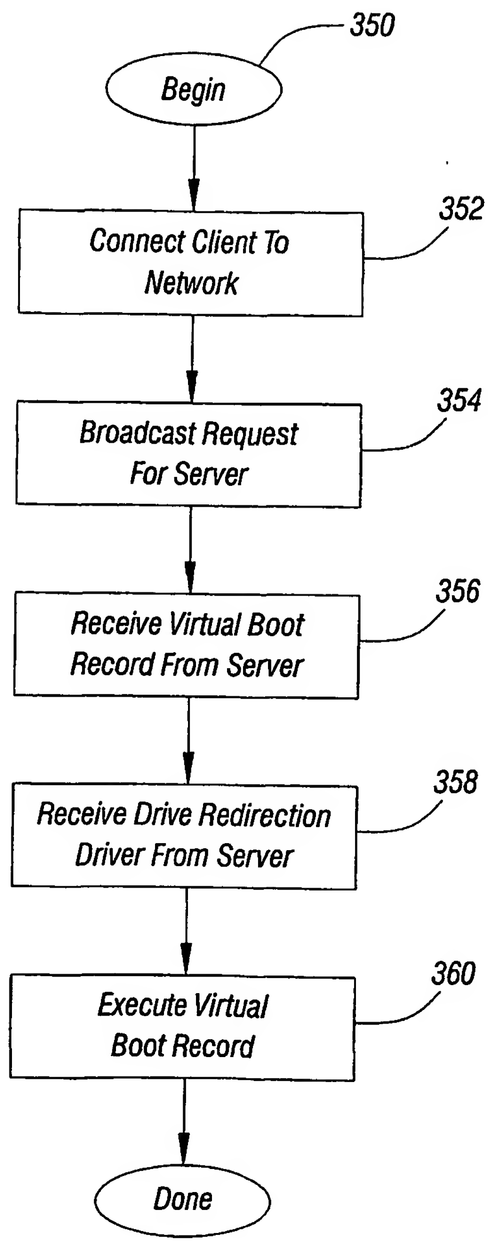

350
Begin
352
Connect Client To Network
354
Broadcast Request For Server
356
Receive Virtual Boot Record From Server
358
Receive Drive Redirection Driver From Server
360
Execute Virtual Boot Record
Done

Figure 7

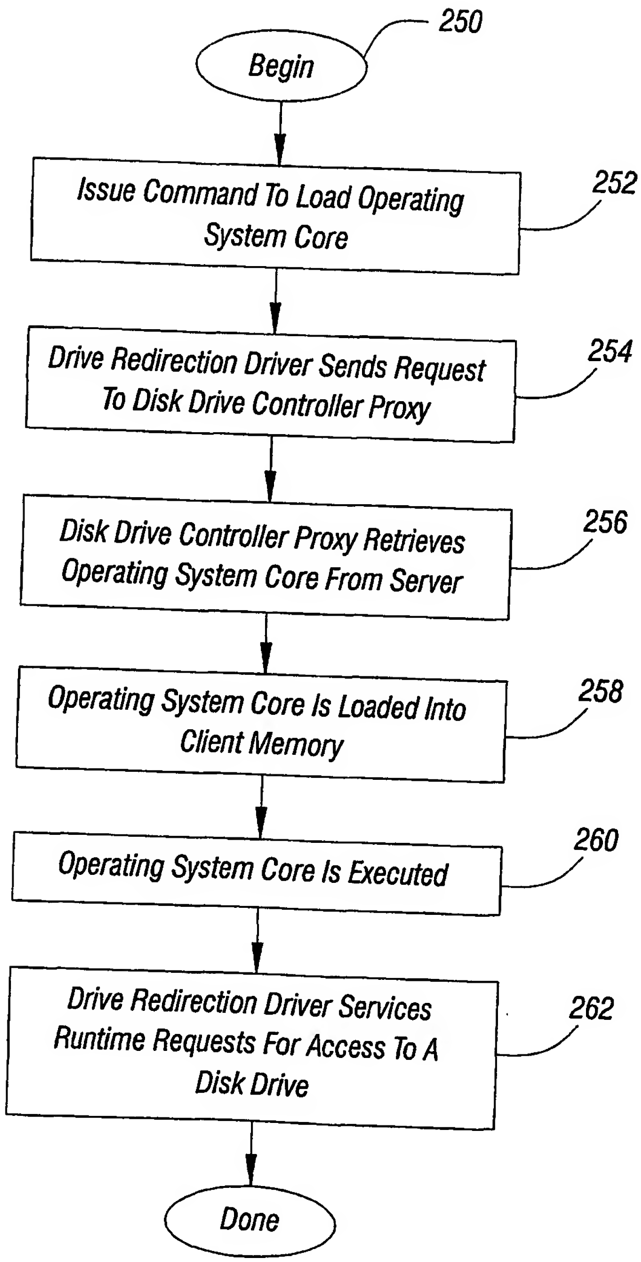

250
Begin
Issue Command To Load Operating System Core
252
Drive Redirection Driver Sends Request To Disk Drive Controller Proxy
254
Disk Drive Controller Proxy Retrieves Operating System Core From Server
256
Operating System Core Is Loaded Into Client Memory
258
Operating System Core Is Executed
260
Drive Redirection Driver Services Runtime Requests For Access To A Disk Drive
262
Done

Figure 8

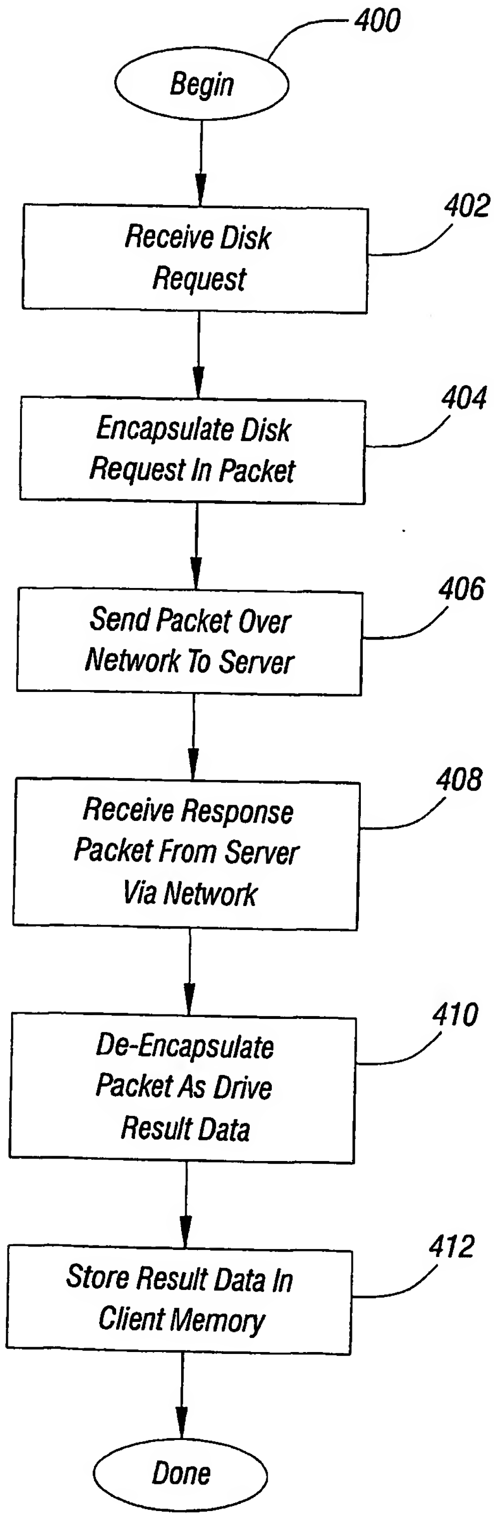

400
Begin
402
Receive Disk Request
404
Encapsulate Disk Request In Packet
406
Send Packet Over Network To Server
408
Receive Response Packet From Server Via Network
410
De-Encapsulate Packet As Drive Result Data
412
Store Result Data In Client Memory
Done

Figure 9

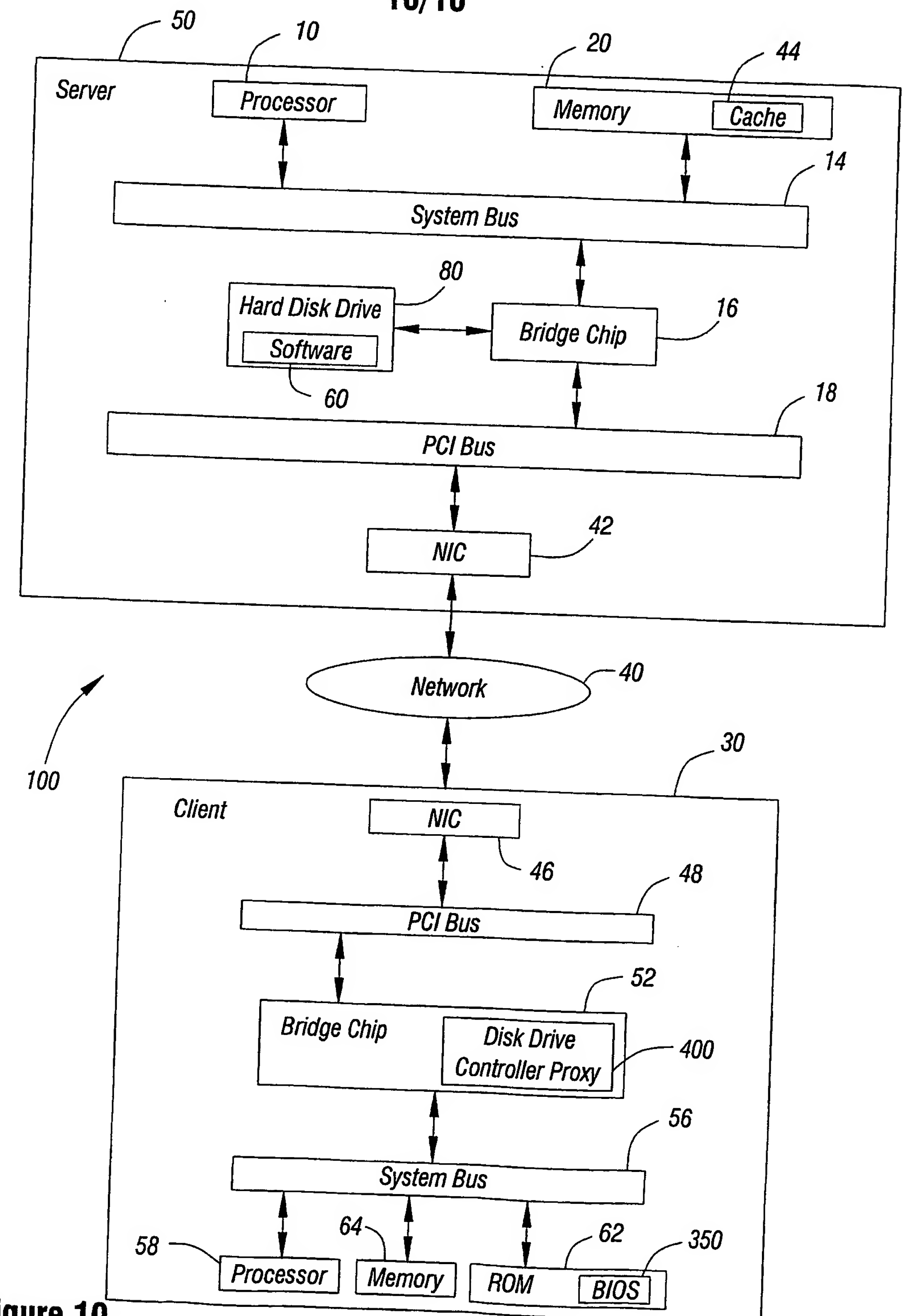

50
10
20
44
Server
Processor
Memory
Cache
14
System Bus
80
Hard Disk Drive
Software
Bridge Chip
16
60
18
PCI Bus
42
NIC
Network
40
30
100
Client
NIC
46
48
PCI Bus
52
Bridge Chip
Disk Drive Controller Proxy
400
56
System Bus
58
64
62
350
Processor
Memory
ROM
BIOS

Figure 10

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 December 2001 (20.12.2001)

PCT

(10) International Publication Number
WO 01/097016 A3

(51) International Patent Classification[7]: G06F 17/30, 3/06

(21) International Application Number: PCT/US01/17263

(22) International Filing Date: 25 May 2001 (25.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/592,882 13 June 2000 (13.06.2000) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **HERNANDEZ, Thomas** [US/US]; 4340 NW 147th Avenue, Portland, OR 97229 (US). **STEWART, David** [US/US]; 8140 SW Aralia Place, Beaverton, OR 97008 (US).

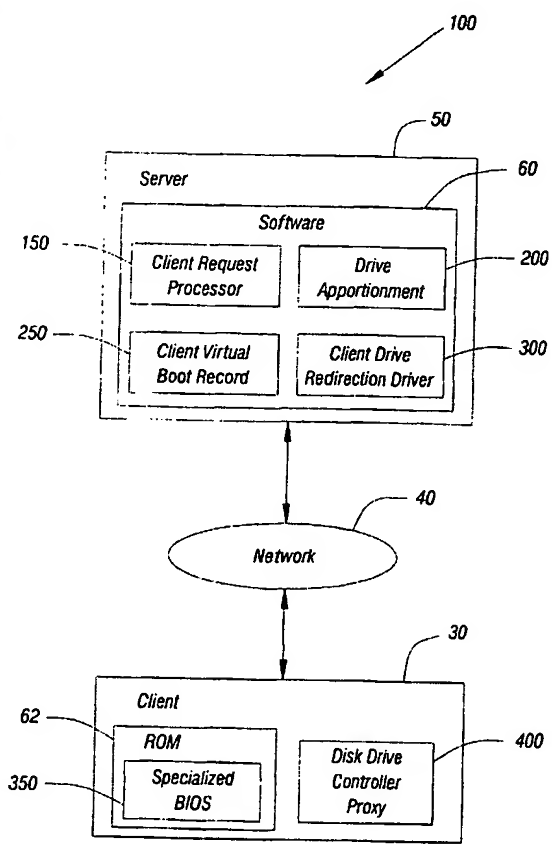
(74) Agent: **TROP, Timothy, N.**; Trop, Pruner & Hu, P.C., 8554 Katy Freeway, Suite 100, Houston, Tx Texas 77024 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: PROVIDING CLIENT ACCESSIBLE NETWORK-BASED STORAGE



(57) Abstract: A system for accessing storage on a server by clients on a network includes software on the server to process requests and allocate storage based on client needs. The clients include a special BIOS and a disk drive controller proxy. The BIOS retrieves operating system software, drivers, and other application software from the server. The disk drive controller proxy routes disk requests to the server for processing. In some embodiments, clients may share storage on the server.

WO 01/097016 A3

patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

(88) Date of publication of the international search report: 10 July 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/17263

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30 G06F3/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

WPI Data, INSPEC, COMPENDEX, IBM-TDB, EPO-Internal, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 611 049 A (PITTS WILLIAM M) 11 March 1997 (1997-03-11)	1,10,23
Y	column 9, line 35 -column 10, line 30; figure 1	17

X	FENG-MING HSIEH ET AL: "A kernel-level DSVM controller for the diskless cluster system" PARALLEL AND DISTRIBUTED SYSTEMS, 1994. INTERNATIONAL CONFERENCE ON HSINCHU, TAIWAN 19-21 DEC. 1994, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, 19 December 1994 (1994-12-19), pages 642-648, XP010223593 ISBN: 0-8186-6555-6	1,10,23
Y	page 642 -page 648	17

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 March 2003	02/04/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bowler, A

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 996 024 A (BLUMENAU STEVEN) 30 November 1999 (1999-11-30) column 2, line 21-32 column 3, line 24-35	17
A	US 5 349 643 A (COX JAMES O ET AL) 20 September 1994 (1994-09-20) column 3, line 43 -column 7, line 41	1,10,16, 17,20,23
A	US 5 758 183 A (SCALES DANIEL J) 26 May 1998 (1998-05-26) column 2, line 54 -column 3, line 38	1,10,17, 23
A	EP 0 752 660 A (SUN MICROSYSTEMS INC) 8 January 1997 (1997-01-08) column 4, line 31 -column 5, line 17	1,10,17, 23

INTERNATIONAL SEARCH REPORT

Inf ation on patent family members

Internati pplication No
PCT/US 01/17263

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
US 5611049	A	11-03-1997	WO	9324890	A1	09-12-1993
			US	6085234	A	04-07-2000
			US	2001016896	A1	23-08-2001
			US	5892914	A	06-04-1999
			US	2001011300	A1	02-08-2001
			AU	2227792	A	30-12-1993
			CA	2136727	A1	09-12-1993
			DE	69227384	D1	26-11-1998
			EP	0643853	A1	22-03-1995
			JP	8500918	T	30-01-1996
US 5996024	A	30-11-1999	NONE			
US 5349643	A	20-09-1994	JP	2574997	B2	22-01-1997
			JP	7200429	A	04-08-1995
US 5758183	A	26-05-1998	NONE			
EP 0752660	A	08-01-1997	US	5802297	A	01-09-1998
			DE	69604734	D1	25-11-1999
			DE	69604734	T2	29-06-2000
			EP	0752660	A1	08-01-1997
			JP	9034825	A	07-02-1997